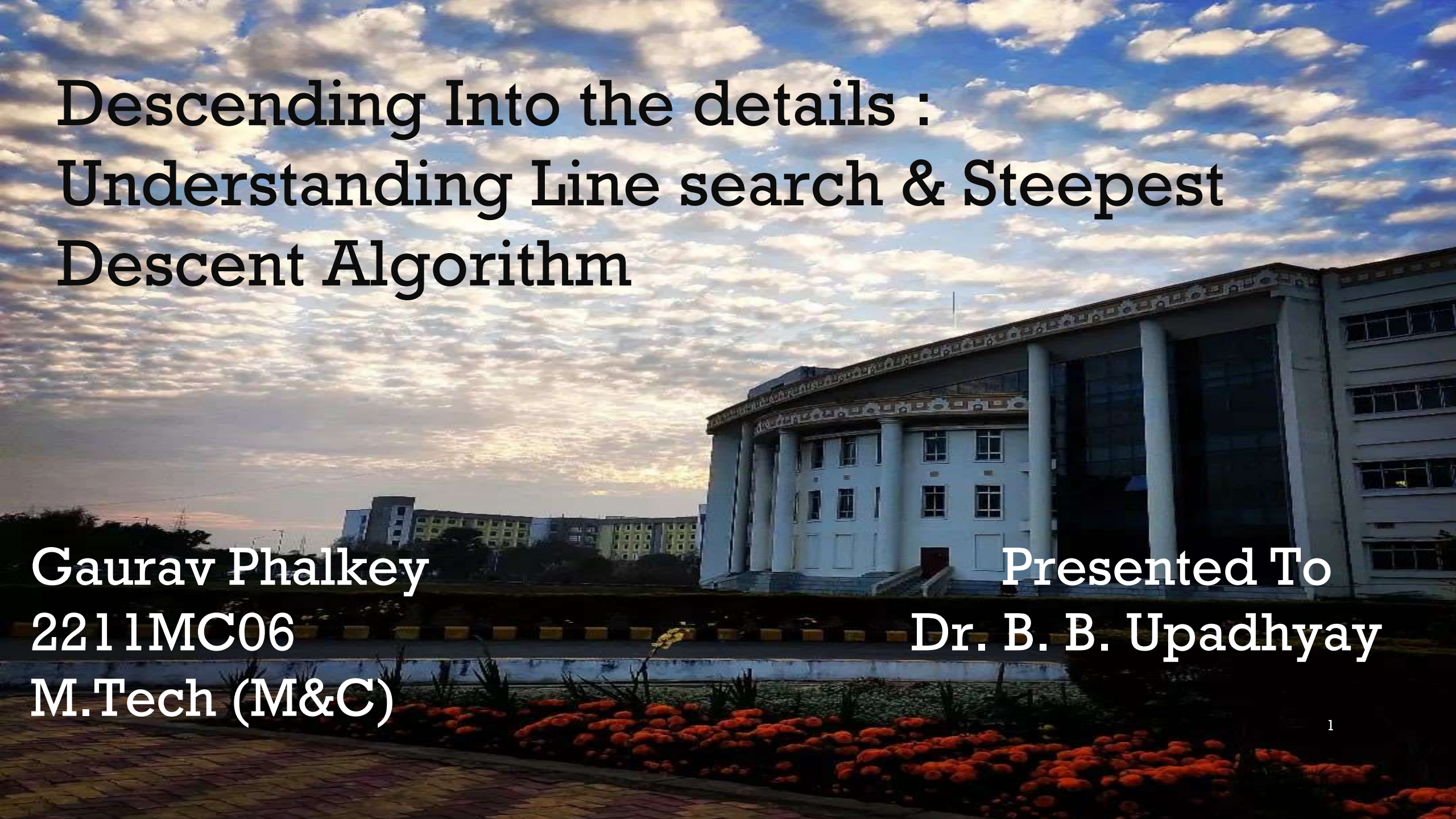
Descending Into the details : Understanding Line search & Steepest Descent Algorithm



Gaurav Phalkey
2211MC06
M.Tech (M&C)

Presented To
Dr. B. B. Upadhyay

What is Gradient ?

- In the case of a univariate function, it is simply the first derivative at a selected point. To check the line segment connecting two function's points lays on or above its curve.
- In the case of a multivariate function, it is a vector of derivatives in each main direction (along variable axes) called partial derivative. To check this function has to be twice differentiable and the Hessian is positive semidefinite

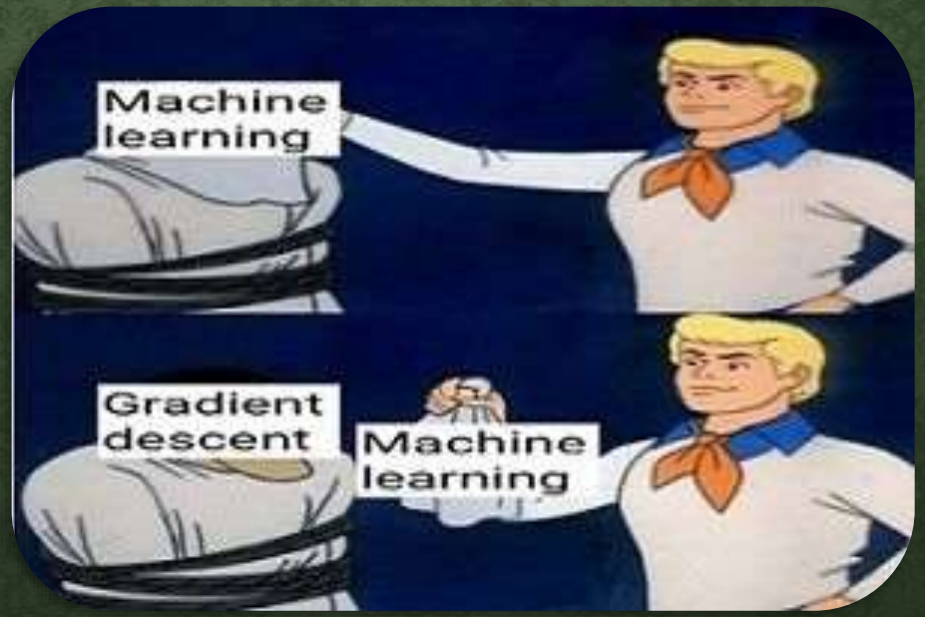
Gradient Descent doesn't work for all Functions !!

A function has to be

- Differentiable

Motivation

Why Do we care so much about Gradient Descent ?



Most ML & Deep Learning techniques involve the minimization of a loss ()$_3$ and this method that serve the purpose

Gradient Descent , Finally !!

- Gradient descent (also often called steepest descent) is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function.
- French Mathematician, Augustin-Louis Cauchy, first suggested it in 1847 .

The main goal is to converge to the minima of the loss functions in the least amount of time. This involves converging fast to the minima in a computationally efficient manner.

Gradient descent, although computationally efficient, provides a slow rate of convergence.

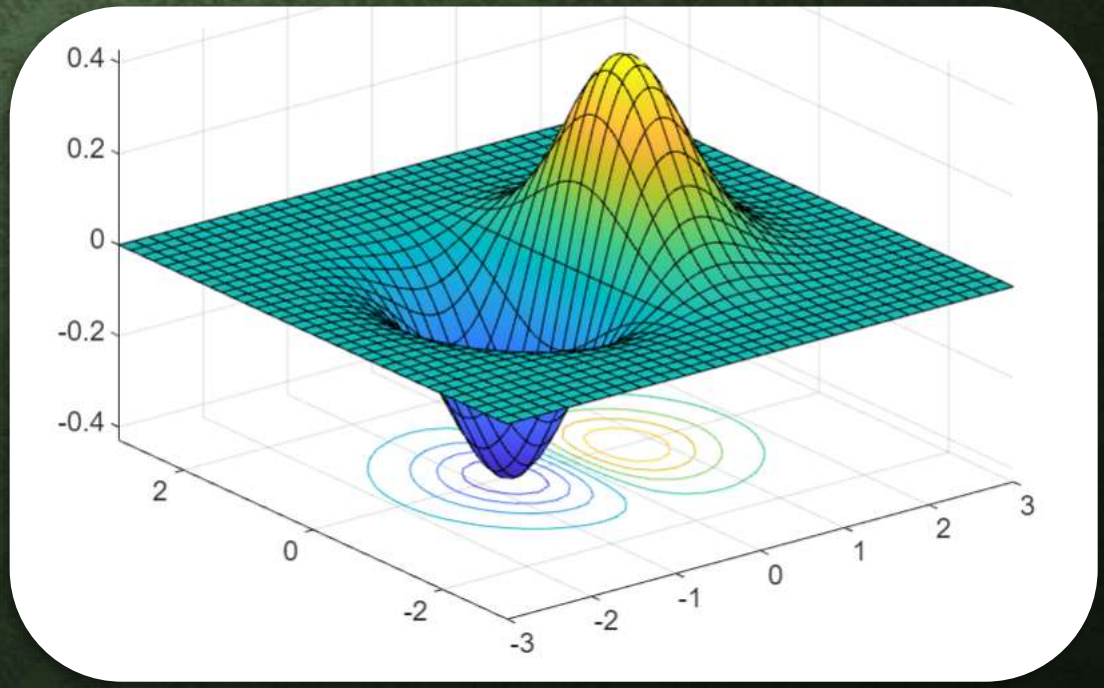
This is where **line search** strategy comes into picture which finds good value of step size & hence better rate of convergence at a slight increase in computational spending.

Unconstrained Minimization

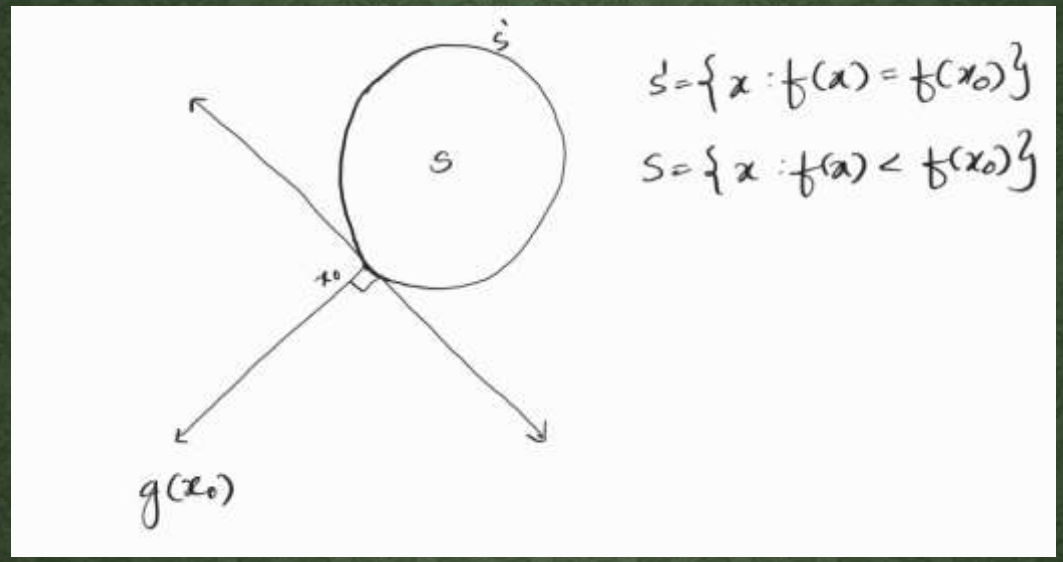
Let $f: R^n \rightarrow R$ Consider the optimization problem

$$\begin{aligned} \text{Min} \quad & f(x) \\ \text{such that} \quad & x \epsilon R^n \end{aligned}$$

$$f(x) = x_1 \exp(-x_1^2 - x_2^2)$$



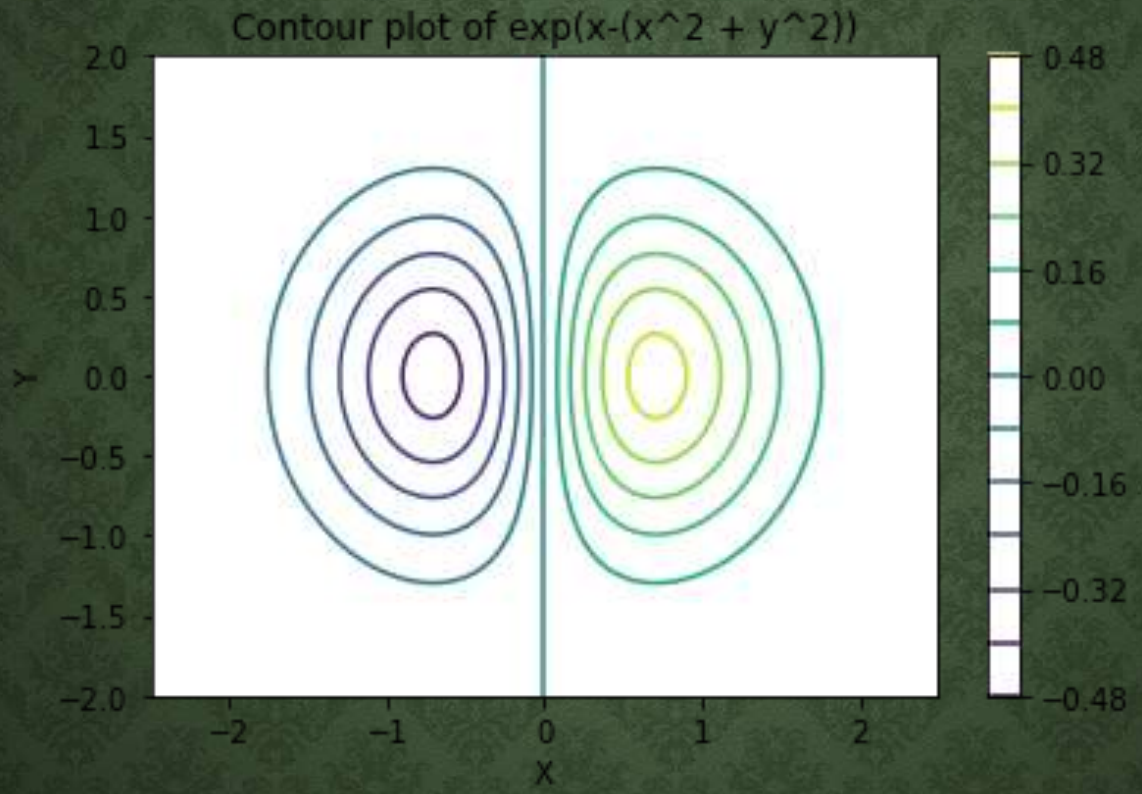
How to check whether direction is Descent direction or not?



Let $x \epsilon R^n$ and $f \epsilon C^1$ and there exist a direction $d \epsilon R^n$, $f(x + \alpha d)$
$g(x) = \nabla f(x)$, if $g(x)^T d < 0$ then d is descent direction of f at x

In other words , any direction d that make obtuse angle with $g(x)$
then that direction d is descent direction

Contour plot of $x_1 \exp(-x_1^2 - x_2^2)$



Value of function decreases if we move towards the inner most concentric circle and increases if we move away from local minimum.

(So it is characteristics of local min point , that function value does not decrease in local neighbourhood of local min point, it either remains constant or increases) .

One Interesting FACT is that Local Minimum is a point where does not exist a Descent direction

It is Nothing but a **FONC** for Unconstrained Minimization

Let $f: R^n \to R,\ f \epsilon C^1$, if x^* is a local minimum of f then $\mathrm{g}(x^*) = 0$

It provide stopping condition for optimisation algorithm

$$\text{Min } f(x) = x_1 \exp(-x_1^2 - x_2^2)$$

$$g(\mathbf{x}) = \begin{pmatrix} \exp(-x_1^2 - x_2^2)(1 - 2x_1^2) \\ \exp(-x_1^2 - x_2^2)(-2x_1 x_2) \end{pmatrix}$$

$$g(x) = 0 \text{ at } \begin{pmatrix} 1/\sqrt{2} \\ 0 \end{pmatrix} \text{ and } \begin{pmatrix} -1/\sqrt{2} \\ 0 \end{pmatrix}$$

Set of stationary points but we don't know which one is local Minima and local maxima.

Need higher order derivative to tell which stationary point is local minima
Which take us to

" SONC "

SONC

Let $f : R^n \rightarrow R$, $f \in C^2$, if x^* is local minimum of f then g(x^*) =0
& H(x^*) is Positive Semi-definite

But wait !!

This Second order Necessary condition are not sufficient
we need something else……

SOSC

Let $f: R^n \rightarrow R$, $f \epsilon C^2$, if $g(x^*) = 0$ and $H(x^*)$ is positive definite
then
x^* is a local minimum of f

$$g(x) = 0 \text{ at } \begin{pmatrix} 1/\sqrt{2} \\ 0 \end{pmatrix} \text{ and } \begin{pmatrix} -1/\sqrt{2} \\ 0 \end{pmatrix}$$

$$H(x_2) = \begin{bmatrix} 2\sqrt{2}\exp(-\frac{1}{2}) & 0 \\ 0 & \sqrt{2}\exp(-\frac{1}{2}) \end{bmatrix} \Rightarrow \text{Positive Definite}$$

hence x_2 is local minimum

$$H(x_1) = \begin{bmatrix} -2\sqrt{2}\exp(-\frac{1}{2}) & 0 \\ 0 & -\sqrt{2}\exp(-\frac{1}{2}) \end{bmatrix} \Rightarrow \text{Negative Definite}$$

hence x_1 is local maximum.

Stopping Conditions for a minimization Problem

$||\, g(x^k)|| = 0$ && $H(x^*)$ is positive semi-definite

But it has some problems too !!

- Second Order Information → Computationally expensive to find but if some extra information is available one can use .

- In Numerical Implementation of algorithm Difficult of maintain norm of $g(x^k)$ to be exactly zero

So It may be a good idea to use some approximate condition like this

$$||\, g(x^k)|| \leq \varepsilon$$

Step length determination

Exact Line search Method : Given a descent direction d^k , determining α^k by solving another optimization problem

$\alpha^{k+1} = \arg\min f(x^k + \alpha\, d^k)$, $\forall > 0$

$\alpha^{k+1} = \arg\min \Phi(\alpha)$, $\forall > 0$

Solving this Optimisation problem for above optimisation problem is computationally expensive

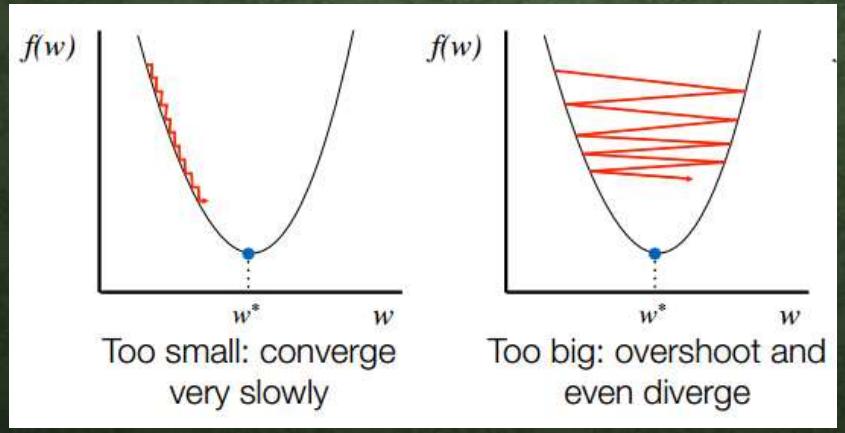
So we move towards Inexact line search method

Inexact Line search Method

An inexact line search will find the largest α among a discrete set of values such that a sufficient decrease condition is achieved

Problems with Inexact line search method is that depends on sequence generated

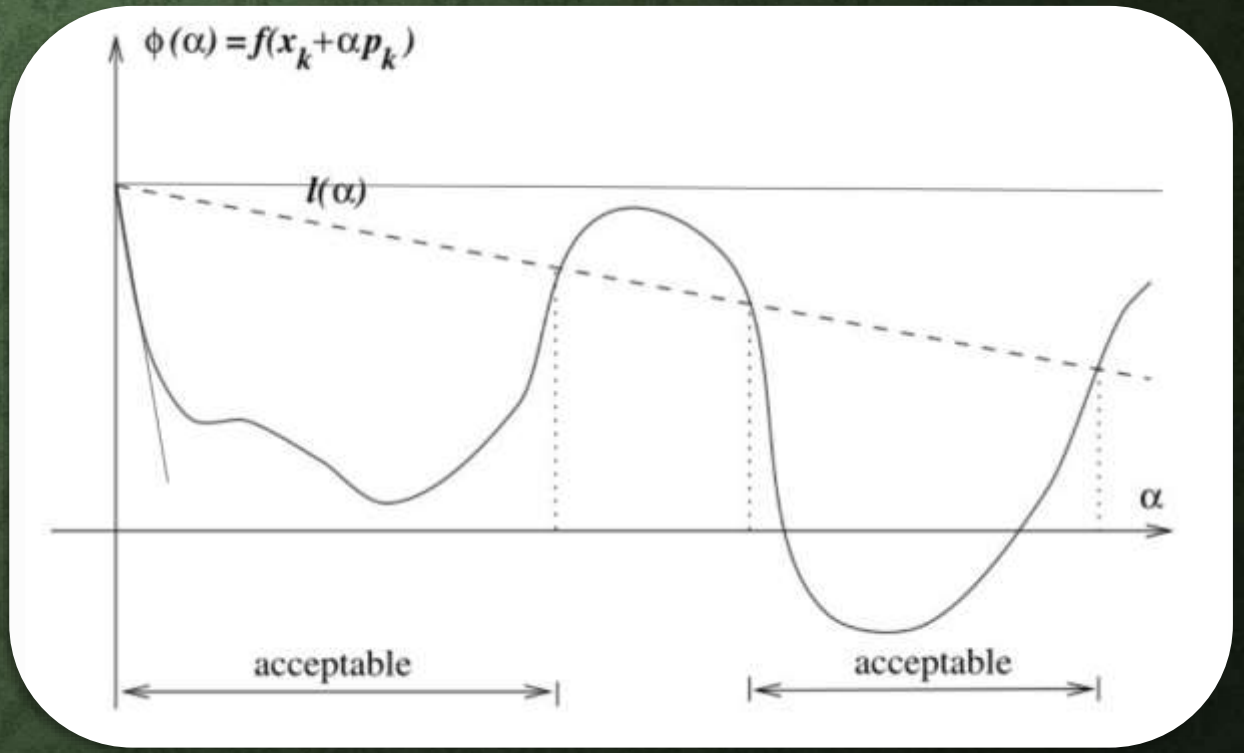
1. Small Decrease in Functional Value relative to step length



2 .Step size are too small comparison to intial rate of decrease of function

Armijo's Condition

- ✓ Armijo's conditions ensures sufficient decrease in functional value
- ✓ To ensure sufficient decrease in functional value , have to choose alpha value such that Decrease in functional value at new alpha is atleast some fraction times decrease in functional value when alpha is 0.

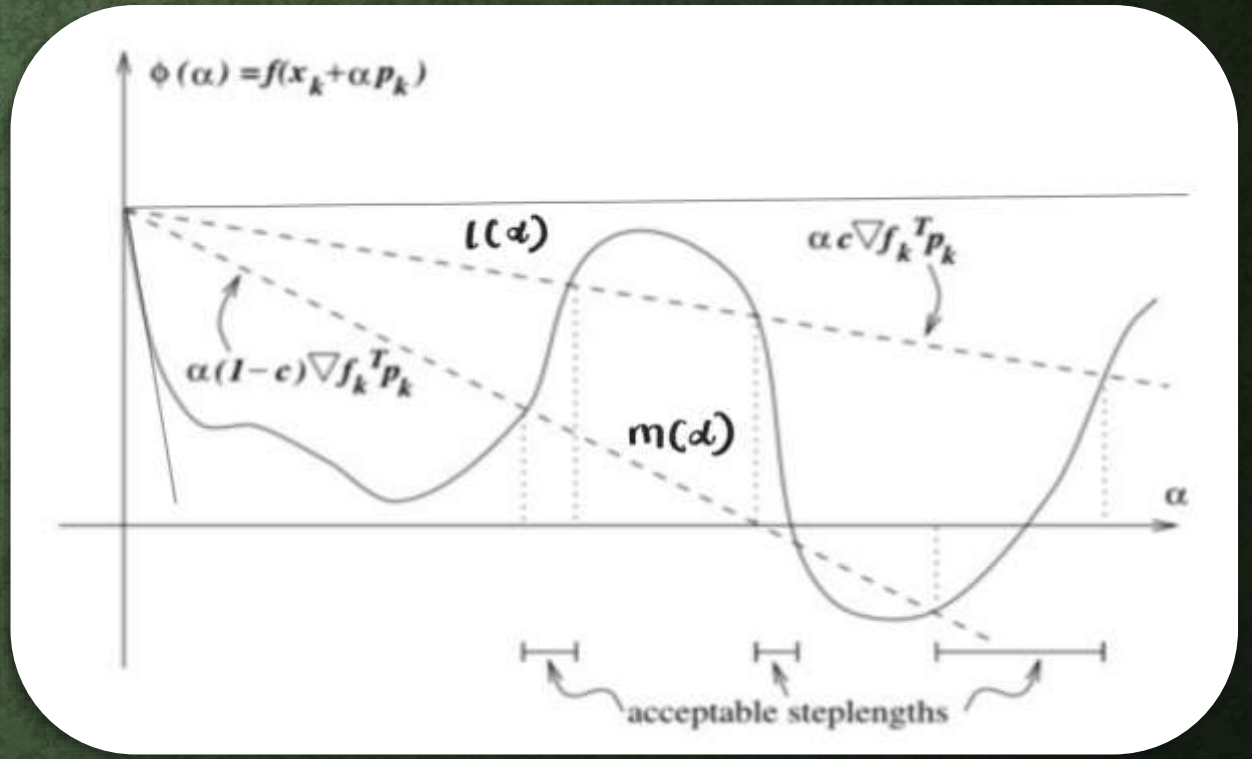


$$l(\alpha) = f(x^k) + c_1\, \alpha g^k p^k \quad , c_1 \in (0,1)$$

Choose α^k such that $f(x^k + \alpha^k p^k) \leq L(\alpha^k)$ ➔ Armijo's Condition

Goldstein' s condition

- ✓ Armijo's Condition does not ensure that step length are sufficient large .
- ✓ To ensure sufficient increase in step length value , have to choose alpha value such that Decrease in functional value at new alpha is atleast some fraction times decrease in functional value when alpha is 0.



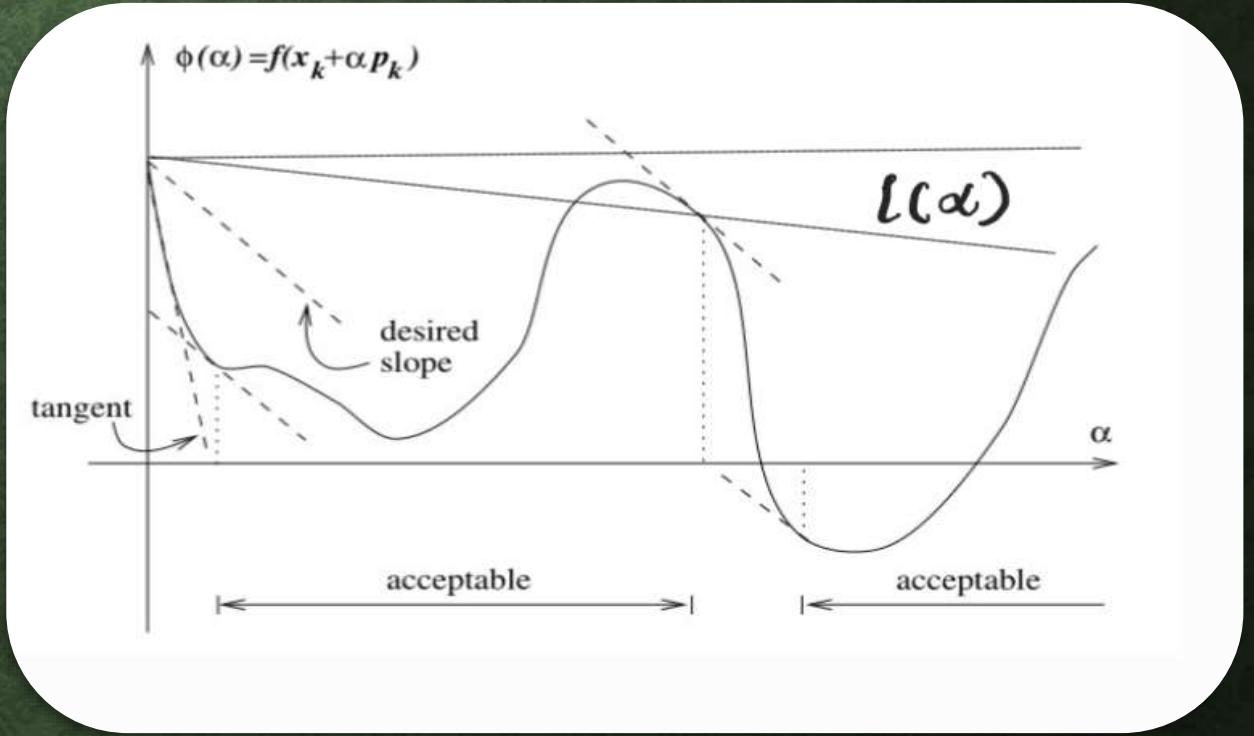
$M(\alpha) = f(x^k) + (1 - c_1)\,\alpha g^k p^k \quad , c_1 \in (0,1)$

Choose α^k such that $f(x^k + \alpha^k p^k) \geq M(\alpha^k)$ ➔ Goldstein 's Condition

Merging two conditions we get ➔ $M(\alpha^k) \leq f(x^k + \alpha^k p^k) \leq L(\alpha^k)$

Wolfe's Condition

- ✓ Wolfe's Suggested that desired slope is some prescribed times(greater than 1) slope when alpha is zero .
- ✓ We are interested in those interval in which functional value has slope in desired range



So the conclusion we got here

Armijo's condition + Goldstein's Condition <<<<< Armijo's Condition + Wolfe's Condition

Finally , Our Steepest Descent algorithm Becomes

1. Initialize x^0 , set k=0
2. WHILE $||\, g(x^k)|| \leq \varepsilon$

   (a). Find descent Direction d for f at x^k

   (b) .Find $\alpha^k > 0$ along d^k such that

   $f(x^k + \alpha^k d^k) < f(x^k)$ && α^k follows Armijo's - Wolfe Conditions

   (c) $x^{k+1} = x^k + \alpha^k d^k$

   (d) k=k+1

END-WHILE

Global Convergence of Line Search Algorithm

By Line search algorithm, $\nabla f(x_k)$ Convergent if $\|\nabla f(x_k)\| \to 0$

$$\lim_{k \to \infty} \|\nabla f(x_k)\| \to 0$$

By Zoutendijk's theorem if line search satisfy Wolfe's Condition then it has to satisfy the angle between search and steepest direction is bounded away by 90 degree

Let ∇f is Lipschitz continuous , α_k is step length θ_k is the angle,

then $\sum_{k=0}^{\infty} cos^2\, \theta_k \, \|\nabla f(x_k)\|^2 < \infty$

The Zoutendijk's conditions above implies that $\lim_{k\to\infty} cos^2\,\theta_k\,\|\nabla f(x_k)\|^2 = 0$, Hence if angle search direction with gradient away by 90 degree then $0 < cos^2\,\theta_k$

By n^{th} divergence test $0 < \varepsilon \leq cos^2\,\theta_k$ for all k

It follows that $\lim_{k\longrightarrow\infty} \|\nabla f(x_k)\| = 0$

Why Gradient descent : Not the best Optimization technique but works best in Neural Network

- On one hand , Gradient Descent is by itself not the most efficient technique, as we use Gradient descent in the beginning ,only later when derivative become small, we start using more sophisticated techniques

- ON the other hand , however, Surprisingly , there is application of Optimization where situation is reverse.

Back-Propagation in Neural Network which is just a algorithmic implementation of Gradient descent works optimally best in comparison to complex optimization technique

How can we explain this paradox ??

The beauty of the solution lies in its simplicity, far simpler than one might imagine.

The problems of machine learning are only in the first crude approximation described as optimization problems.

In reality, the desired solution should not be too close to the actual minimum of the objective function which is generally Loss function

Desired value of the objective function is supposed to be proportional to the number of training samples and to the accuracy required.

Since in machine learning, we do not need to get too close to the actual minimum, we thus do not need to switch from fast gradient descent to any other technique – and this explains why gradient descent is best in business

Supremacy of Gradient Descent Still Prevails !!

Batch Gradient Descent

- Batch gradient descent sums the error for each point in a training set, updating the model only after all training examples have been evaluated. This process referred to as a training epoch.

- While this batching provides computation efficiency, it can still have a long processing time for large training datasets as it still needs to store all of the data into memory.

- One of the main drawbacks of batch gradient descent is that it can converge to a local minimum instead of the global minimum. This is because the algorithm updates the model parameters based on the average of the gradients of all examples in the batch.

Mini Batch Gradient Descent

- Mini-batch gradient descent combines concepts from both batch gradient descent and stochastic gradient descent.

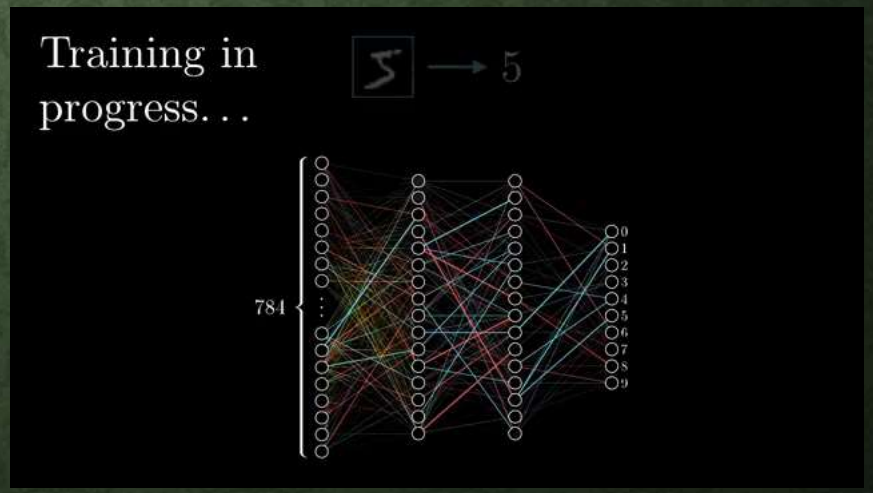
- It splits the training dataset into small batch sizes & performs updates on each batches.

- This approach strikes a balance between the computational efficiency of batch gradient descent and the speed of stochastic gradient descent

Stochastic gradient descent

It updates the model parameters based on the gradient of a single example at a time, rather than a batch of examples like in batch gradient descent.

- Since only one training example is processed at a time, it is easier to store in memory
- It is more computationally efficient.



References

- Smith, A.M. (2018). Why Gradient Descent – Not the Best Optimization Technique – Works Best in Neural Networks: Qualitative Explanation. arXiv preprint arXiv:1803.04781

- Dai Z, Wen F. Global convergence of a modified Hestenes-Stiefel nonlinear conjugate gradient method with Armijo line search. Numerical Algorithms. 2012 Jan;59(1):79-93

- https://optimization.cbe.cornell.edu/index.php?title=Line_search_methods#cite_note-nocedal-1%20paper

Thank You all for being here !!

“ Steepest descent : Where every step is a step towards a better tomorrow , unless you step too far “